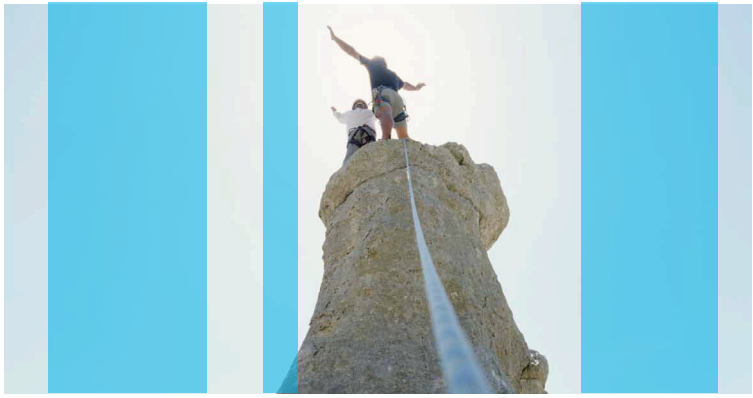
Software Improvement Group



Assessing and Predicting Reliability of Applications

Tobias Kuipers, CTO, SIG

Spider session November 2, 2010

29 October 2010 Rembrandttoren, 14e etage
Amstelplein 1
1096 HA Amsterdam
info@sig.eu
www.sig.eu

SIG helps improve software

Intro SIG

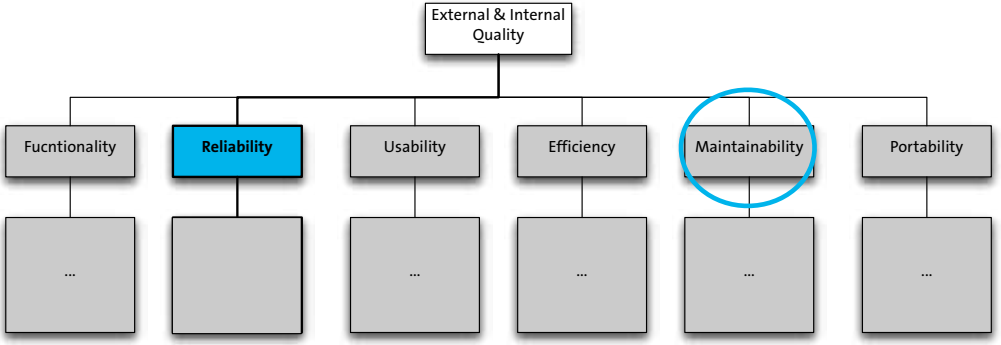
- Started in 2000 from Centre for Mathematics and Computer Science (CWI)
- Now helping clients with Application Portfolio Engineering
- Assess the state of the current Application Portfolio
- Cost, risk, environmental impact, ...
- Deliver roadmap to optimize Application Portfolio

- New area: Reliability

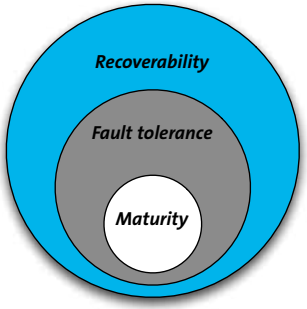
- Need way to quickly assess or predict reliability of an application

We build quality models based on ISO 9126 to support our work

Software Improvement Group



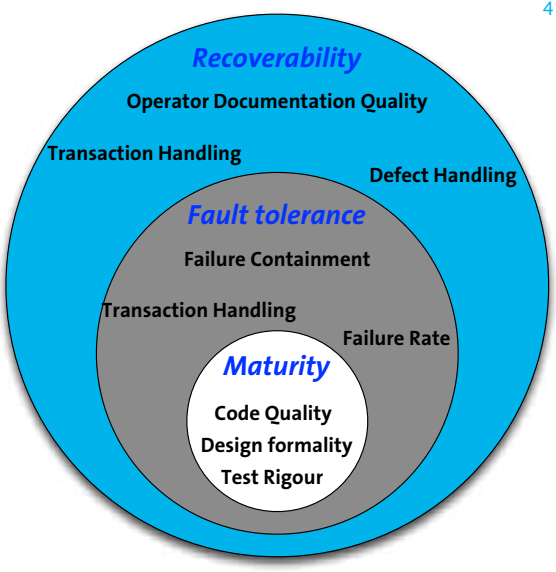
TÜViT



Reliability Model

Reliability [ISO 9126]

- the capability of the software product to maintain **a specified level of performance** when used under **specified conditions**



Reliability Model

Reliability

- the capability of the software product to maintain **a specified level of performance** when used under

Maturity

- the ability of th… product to **av…** a results… software

Fault tolerance

… in cases of software faults or infringement of its specified interface

Recoverability

- the ability of the so… product to **re-estab… specified level of pe… and recover** the dat… affected in case of fa…

How many bugs do we have?

Can we keep going if a bug manifests itself?

If we break down, how long before we are back up?

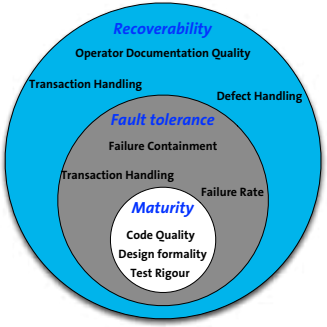
	Code Quality (CQ)	Test Rigor (TR)	Transaction Handli… (TH)	Failure Containment (FC)	Operator Documentation Quality (ODQ)	Design Formality (DF)	Failure Rate (FR)	Defect Handling (DH)
Maturity	X	X				X		
Fault tolerance			X	X			X	
Recoverability			X		X			X

Reliability model built up from measurable components

Used to predict reliability of a system

Code Quality

- The number of bugs per 1000 lines of code, as determined by an industry standard bug finding tool for this particular environment
- FindBugs, Coverity, Parasoft, DevEnterprise,...
- The less bugs, the higher the rating



Design Formality

- Level of confidence in transferring the validated design to the actual code
- Is a reliability model created for this system?
- Is there traceability of the elements in the model to the elements in the code?
- Has the code been proven to demonstrate the properties of the model?
- Is the model periodically improved to reflect the increasing knowledge about the reliability properties of the actual system?

- Higher rating for each of the steps demonstrably fulfilled

More measurable components

Transaction Handling

- System is either
  - non-transactional
  - lightweight transactional data
    - only core data
  - fully transactional data
    - all data
  - fully transactional
    - all data and program state
- More transactional is a higher rating

Failure Containment

- Start with a reconstructed architectural view
- Or use the original reliability analysis model if it is known to be accurate
- Determine “critical components”
- Determine level of fault propagation from one component to the next
- The lower the fault propagation, the higher the rating

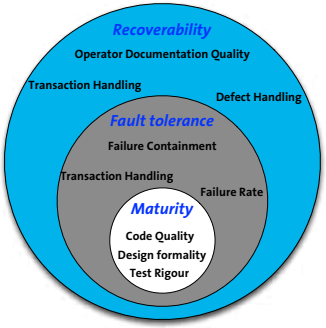
More measurable components

Operator Documentation Quality

- How much effort does it take to restart the system
- Is there a clear instruction on how to recover:
  - no
  - no instructions, but lists all operating environments
  - lists environments and instruction to restart entire system
  - above plus instructions to restart individual component
  - above plus diagnostic procedure to identify cause of failure
- More steps is a higher rating

Failure Rate

- Mean time between failure (MTBF)
- Higher MTBF is a higher rating



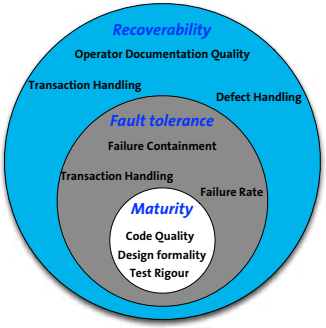
More measurable components

Test Rigor

- The amount of branch coverage and statement coverage while testing
- Test process must demonstrate the relation between test and amount of code that is covered by the test
- Higher rating for more branch coverage and statement coverage. Statement coverage more important than branch coverage

Defect Handling

- The amount of time needed to repair a defect.
- Higher rating for less time to fix a defect



Aggregating the results

	Code Quality (CQ)	Test Rigor (TR)	Transaction Handling (TH)	Failure Containment (FC)	Operator Documentation Quality (ODQ)	Design Formality (DF)	Failure Rate (FR)	Defect Handling (DH)	
	★★☆☆☆	★★☆☆☆	★★★☆☆	★★★★☆	★☆☆☆☆	★★★★☆	★★☆☆☆	★★☆☆☆	
Maturity	X	X				X			★★★☆☆
Fault tolerance			X	X			X		★★★☆☆
Recoverability			X		X			X	★★☆☆☆

Rating & Aggregation
Examples

Test Rigour

Decision Coverage	Line Coverage (%)					
	0 – 10	10 – 25	25 – 60	60 – 80	80 – 90	90 –
0 – 10	0.5	1	1.5	2	2.5	3
10 – 30	1	1.5	2	2.5	3	3.5
30 – 50	1.5	2	2.5	3	3.5	4
50 – 70	2	2.5	3	3.5	4	4.5
70 – 90	2.5	3	3.5	4	4.5	5
90 – 100	3	3.5	4	4.5	5	5.5

Defect Handling

0 days 28 days 70 days 182 days >182 days

An example case

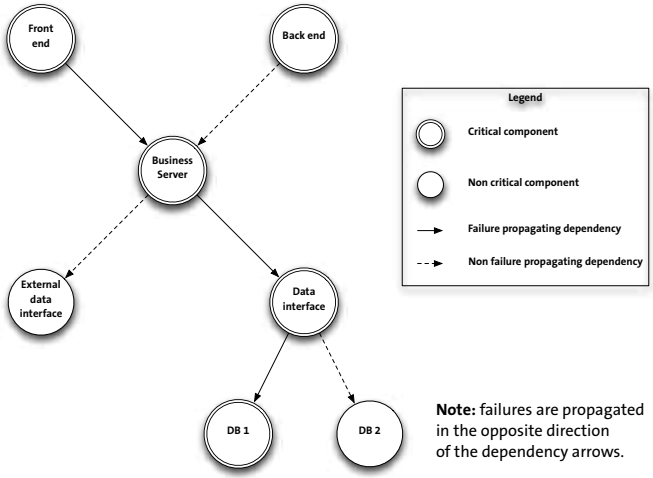
System

- multi-user insurance payment system
- used by both customers and employees

Technical aspects

- main technologies: Java, JSP
- volume:
  - 150 KLOC Java (production)
  - 80 KLOC Java (test)
  - 15 KLOC JSP

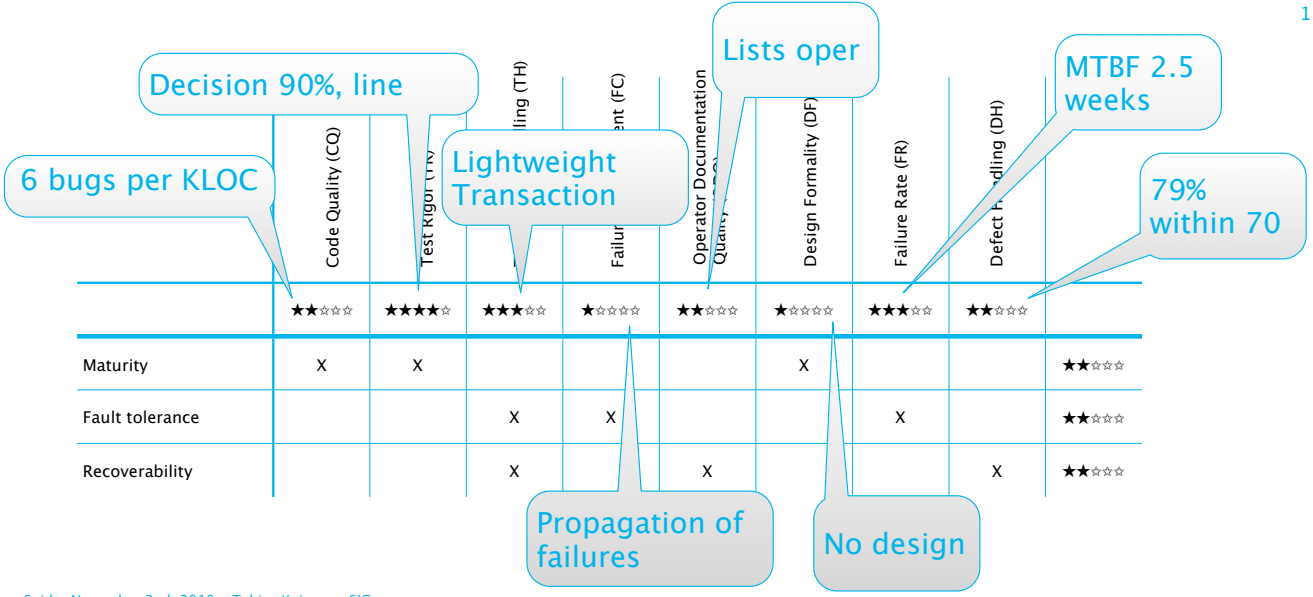
Architecture View



Note: failures are propagated in the opposite direction of the dependency arrows.

Aggregating the results

Software Improvement Group



Questions

Tobias Kuipers
t.kuipers@sig.eu
06-14488847